1. **Понятие модели. Виды моделей.**

Под **моделью** будем понимать такой материальный или мысленно представляемый объект, который в процессе исследования заменяет собой объект-оригинал таким образом, что его непосредственное изучение дает новые сведения об объекте-оригинале.

1. Линейное программирование - линейное преобразование переменных в системах линейных уравнений. Сюда можно отнести: симплекс-метод, распределительный метод, статический матричный метод решения материальных баллансов.

2. Дискретное программирование представлено двумя классами методов: локализационные и комбинаторные методы. К локализационным относятся методы линейного целочисленного программирования. К комбинаторным, например, метод ветвей и границ.

3. Математическая статистика используется для корреляционного, регресионного и дисперсионного анализа экономических процессов и явлений. Корреляционный анализ применяется для установления тесноты связи между двумя или более стохастически независимыми процессами или явлениями. Регрессионный анализ устанавливает зависимость случайной величины от неслучайного аргумента. Дисперсионный анализ - установление зависимости результатов наблюдений от одного или нескольких факторов в целях выявления важнейших.

4. Динамическое программирование используется для планирования и анализа экономических процессов во времени. Динамическое программирование представляется в виде многошагового вычислительного процесса с последовательной оптимизацией целевой функции. Некоторые авторы относят сюда же имитационное моделирование.

5. Теория игр представляется совокупностью методов, используемых для определения стратегии поведения конфликтующих сторон.

6. Теория массового обслуживания - большой класс методов, где на основе теории вероятностей оцениваются различные параметры систем, характеризуемых как системы массового обслуживания.

7. Теория управления запасами объединяет в себе методы решения задач, в общей формулировке сводящихся к определению рационального размера запаса какой-либо продукции при неопределенном спросе на нее.

8. Стохастическое программирование. Здесь исследуемые параметры являются случайными величинами.

9. Нелинейное программирование относится к наименее изученному, применительно к экономическим явлениям и процессам, математическому направлению.

10. Теория графов - направление математики, где на основе определенной символики представляется формальное описание взаимосвязанности и взаимообусловленности множества элементов (работ, ресурсов, затрат и т.п.). До настоящего времени наибольшее практическое применение получили так называемые сетевые графики.

1. **Классификация моделей**
2. По целевому назначению модели можно делить на:

* модели структуры
* модели функционирования
* стоимостные модели.

***Модели структуры*** отображают связи между компонентами объекта и внешней средой и подразделяются на:

* *каноническую модель*, характеризующую взаимодействие объекта с окружением через входы и выходы;
* *модель внутренней структуры*, характеризующую состав компонентов объекта и связи между ними;
* модель иерархической структуры (дерево системы), в которой объект (целое) расчленяется на элементы более низкого уровня, действия которых подчинены интересам целого.

Модель структуры обычно представляется в виде блок-схемы, реже графов и матриц связей.

***Модели функционирования*** включают широкий спектр символических моделей, например:

* **модель жизненного цикла системы,** описывающая процессы существования системы от зарождения замысла ее создания до прекращения функционирования**;**
* **модели операций,** выполняемых объектом и представляющих описание взаимосвязанной совокупности процессов функционирования отдельных элементов объекта при реализации тех или иных функций объекта.
* **информационные модели,** отображающие во взаимосвязи источники и потребители информации, виды информации, характер ее преобразования, а также временные и количественные характеристики данных;
* **процедурные модели,** описывающие порядок взаимодействия элементов исследуемого объекта при выполнении различных операций, например обработки материалов, деятельности персонала, использования информации, в том числе и реализации процедур принятия управленческих решений;
* **временные модели,** описывающие процедуру функционирования объекта во времени и распределение ресурса «время» по отдельным компонентам объекта.

***Стоимостные модели,*** как правило, сопровождают модели функционирования объекта и по отношению к ним вторичны, «питаются» от них информацией и совместно с ними позволяют проводить комплексную технико-экономическую оценку объекта или его оптимизацию по экономическим критериям.

2. По характеру отражения причинно-следственных связей:

* детерминированные;
* недетерминированные (вероятностные, стохастические), учитывающие фактор неопределённости.

3. По способу отражения фактора времени:

* статические. Здесь все зависимости относятся к одному моменту или периоду времени);
* динамические, характеризующие изменения процессов во времени.

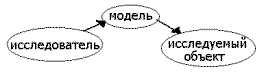
4. По форме математических зависимостей:

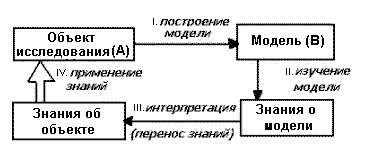
* линейные. Наиболее удобны для анализа и вычислений, вследствие чего получили большое распространение;
* нелинейные.

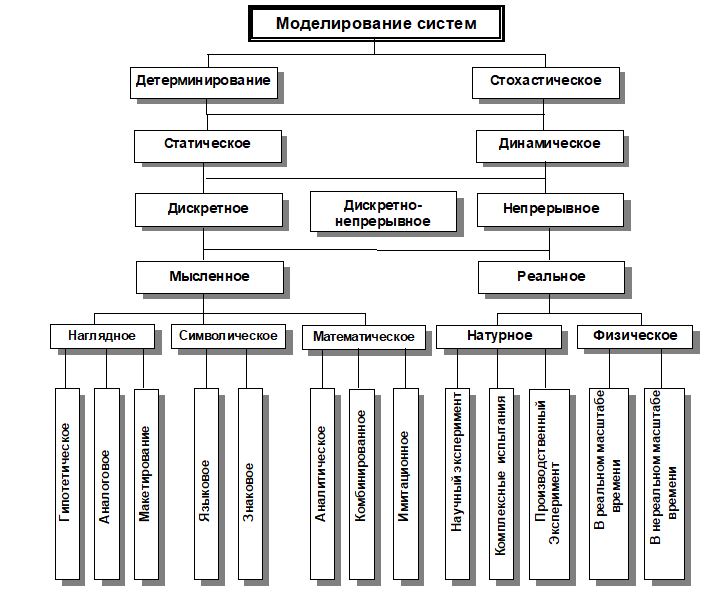
**3. Понятие моделирования. Виды моделирования.**

**Моделирование**, в таком случае, представляет собой процесс построения, изучения и применения моделей.

Процесс моделирования включает в себя три элемента: субъект исследования (исследователь), объект исследования, модель. Ситуацию иллюстрирует рисунок.



**Сущность процесса моделирования** иллюстрирует схема, представленная на рисунке

Моделирование - циклический процесс. Это означает, что за первым четырехшаговым циклом может последовать второй, третий и т.д. При этом знания об исследуемом объекте расширяются и уточняются, а исходная модель постепенно совершенствуется.

***Детерминированное моделирование*** отображает детерминированные процессы, т.е. процессы, в которых предполагается отсутствие всяких случайных воздействий.

***Стохастическое моделирование*** отображает вероятностные процессы и события. В этом случае анализируется ряд реализаций случайного процесса и оцениваются средние характеристики, т.е. набор однородных реализаций.

***Статическое моделирование*** служит для описания поведения объекта в какой-либо момент времени.

***Динамическое моделирование*** отражает поведение объекта во времени.

***Дискретное моделирование*** служит для описания процессов, которые предполагаются дискретными. У дискретных процессов состояния четко отделены друг от друга, и для каждого состояния можно указать соседние

***Непрерывное моделирование*** позволяет отразить непрерывные процессы в системах. У непрерывных процессов между двумя любыми состояниями всегда можно выделить промежуточные, из чего следует, что понятие соседних состояний к такому процессу неприменимо

***Дискретно-непрерывное моделирование*** используется для тех случаев, когда хотят выделить наличие как дискретных, так и непрерывных процессов.

***Мысленное моделирование*** часто является единственным способом моделирования объектов, которые либо практически нереализуемы в заданном интервале времени, либо существуют вне условий, возможных для их физического создания.

При ***реальном моделировании*** используется возможность исследования различных характеристик либо на реальном объекте целиком, либо на его части.

При ***наглядном моделировании*** на базе представлений человека о реальных объектах создаются различные наглядные модели, отображающие явления и процессы, протекающие в объекте.

***Символическое моделирование*** представляет собой искусственный процесс создания логического объекта, который замещает реальный и выражает основные свойства его отношений с помощью определенной системы знаков и символов.

Под **математическим моделированием** будем понимать процесс установления соответствия данному реальному объекту некоторого математического объекта, называемого математической моделью, и исследование этой модели, позволяющее получать характеристики рассматриваемого реального объекта.

В основу ***гипотетического моделирования*** исследователем закладывается некоторая гипотеза о закономерностях протекания процесса в реальном объекте, которая отражает уровень знаний исследователя об объекте и базируется на причинно-следственных связях между входом и выходом изучаемого объекта. Гипотетическое моделирование используется, когда знаний об объекте недостаточно для построения формальных моделей.

***Аналоговое моделирование*** основывается на применении аналогий различных уровней. Наивысшим уровнем является полная аналогия, имеющая место только для достаточно простых объектов. С усложнением объекта используют аналогии последующих уровней, когда аналоговая модель отображает несколько либо только одну сторону функционирования объекта.

Существенное место при мысленном наглядном моделировании занимает ***макетирование***.Мысленный макет может применяться в случаях, когда протекающие в реальном объекте процессы не поддаются физическому моделированию, либо может предшествовать проведению других видов моделирования. В основе построения мысленных макетов также лежат аналогии, однако обычно базирующиеся на причинно-следственных связях между явлениями и процессами в объекте.

Если ввести условное обозначение отдельных понятий, т.е. знаки, а также определенные операции между этими знаками, то можно реализовать ***знаковое моделирование*** и с помощью знаков отображать набор понятий – составлять отдельные цепочки из слов и предложений. Используя операции объединения, пересечения и дополнения теории множеств, можно в отдельных символах дать описание какого-то реального объекта.

В основе ***языкового моделирования*** лежит некоторый тезаурус. Последний образует из наборов входящих понятий, причем этот набор должен быть фиксированным. Следует отметить, что между тезаурусом и обычным словарем имеются принципиальные различия. Тезаурус – словарь, который очищен от неоднозначности, т.е. в нем каждому слову может соответствовать лишь единственное понятие, хотя в обычном словаре одному слову могут соответствовать несколько понятий.

Для ***аналитического*** моделирования характерно то, что процессы функционирования элементов системы записываются в виде некоторых функциональных соотношений (алгебраических, интегродифференциальных, конечно-разностных и т.п.) или логических условий.

***Аналитическая модель*** может быть исследована следующими методами:

а) аналитическим, когда стремятся получить в общем виде явные зависимости для искомых характеристик;

б) численным, когда, не умея решать уравнения в общем виде, стремятся получить числовые результаты при конкретных начальных данных;

в) качественным, когда, не имея решения в явном виде, можно найти некоторые свойства решения (например, оценить устойчивость решения).

При ***имитационном моделировании*** реализующий модель алгоритм воспроизводит процесс функционирования системы ***S*** во времени, причем имитируются элементарные явления, составляющие процесс с сохранением их логической структуры и последовательности протекания во времени, что позволяет по исходным данным получить сведения о состояниях процесса в определенные моменты времени, дающие возможность оценить характеристики системы ***S.***

***Комбинированное (аналитико-имитационное) моделирование*** при анализе и синтезе систем позволяет объединить достоинства аналитического и имитационного моделирования. При построении комбинированных моделей проводится предварительная декомпозиция процесса функционирования объекта на составляющие подпроцессы и для тех из них, где это возможно, используются аналитические модели.

***Натурное моделирование***, под которым понимают проведение исследования на реальном объекте с последующей обработкой результатов эксперимента на основе теории подобия.

***Физическое моделирование*** отличается от натурного тем, что исследование проводится на установках, которые сохраняют природу явлений и обладают физическим подобием.

**4. Принципы системного подхода в моделировании.**

В настоящее время при анализе и синтезе сложных (больших) систем получил развитие системный подход, который отличается от классического (или индуктивного) подхода. Cистемный подход предполагает последовательный переход от общего к частному, когда в основе рассмотрения лежит цель, причем исследуемый объект выделяется из окружающей среды.

**Объект моделирования.** Система S – целенаправленное множество взаимосвязанных элементов любой природы. Внешняя среда Е – множество существующих вне системы элементов любой природы, оказывающих влияние на систему или находящихся под ее воздействием.

В зависимости от цели исследования могут рассматриваться разные соотношения между самим объектом S и внешней средой Е. Таким образом, в зависимости от уровня, на котором находится наблюдатель, объект исследования может выделяться по-разному и могут иметь место различные взаимодействия этого объекта с внешней средой.

При системном подходе к моделированию систем необходимо, прежде всего, четко определить цель моделирования. Поскольку невозможно полностью смоделировать реально функционирующую систему (системуоригинал, или первую систему), создается модель (система-модель, или вторая система) под поставленную проблему. Таким образом, применительно к вопросам моделирования цель возникает из требуемых задач моделирования, что позволяет подойти к выбору критерия и оценить, какие элементы войдут в создаваемую модель М. Поэтому необходимо иметь критерий отбора отдельных элементов в создаваемую модель.

Важным для системного подхода является определение структуры системы – совокупности связей между элементами системы, отражающих их взаимодействие. Структура системы может изучаться извне с точки зрения состава отдельных подсистем и отношений между ними, а также изнутри, когда анализируются отдельные свойства, позволяющие системе достигать заданной цели, т.е. когда изучаются функции системы.

В соответствии с этим наметился ряд подходов к исследованию структуры системы с ее свойствами, к которым следует прежде всего отнести структурный и функциональный.

Правильное понимание взаимосвязей как внутри самой модели M, так и взаимодействия ее с внешней средой Е в значительной степени определяется тем, на каком уровне находится наблюдатель. Простой подход к изучению взаимосвязей между отдельными частями модели предусматривает рассмотрение их как отражение связей между отдельными подсистемами объекта. Такой классический подход может быть использован при создании достаточно простых моделей.

В основе системного подхода лежит рассмотрение системы как интегрированного целого, причем это рассмотрение при разработке начинается с главного – формулировки цели функционирования. На основе исходных данных Д , которые известны из анализа внешней системы, тех ограничений, которые накладываются на систему сверху либо исходя из возможностей ее реализации, и на основе цели функционирования формулируются исходные требования Т к модели системы S . На базе этих требований формируются ориентировочно некоторые подсистемы П , элементы Э и осуществляется наиболее сложный этап синтеза – выбор В составляющих системы, для чего используются специальные критерии выбора КВ . При моделировании необходимо обеспечить максимальную эффективность модели системы. Эффективность обычно определяется как некоторая разность между какими-то показателями ценности результатов, полученных в итоге эксплуатации модели, и теми затратами, которые были вложены в ее разработку и создание.

**Стадии разработки моделей.** На базе системного подхода может быть предложена и некоторая последовательность разработки моделей, когда выделяют две основные стадии проектирования: макропроектирование и микропроектирование.

На стадии макропроектирования на основе данных о реальной системе S и внешней среде Е строится модель внешней среды, выявляются ресурсы и ограничения для построения модели системы, выбирается модель системы и критерии, позволяющие оценить адекватность модели М реальной системы S.

Построив модель системы и модель внешней среды на основе критерия эффективности функционирования системы в процессе моделирования выбирают оптимальную стратегию управления, что позволяет реализовать возможности модели по воспроизведению отдельных сторон функционирования реальной системы S .

Стадия микропроектирования в значительной степени зависит от конкретного типа выбранной модели. В случае имитационной модели необходимо обеспечить создание информационного, математического, технического и программного обеспечения системы моделирования. На этой стадии можно установить основные характеристики созданной модели, оценить время работы с ним и затраты ресурсов для получения заданного качества соответствия модели процессу функционирования системы S. Независимо от типа используемой модели М при ее построении необходимо руководствоваться рядом принципов системного подхода:

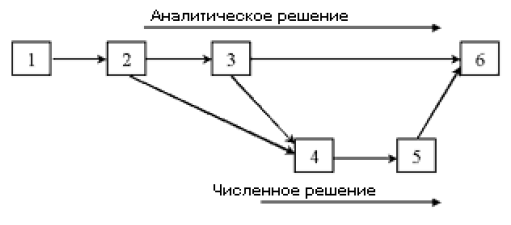
1) пропорционально-последовательное продвижение по этапам и направлениям создания модели;

2) согласование информационных, ресурсных, надежностных и других характеристик;

3) правильное соотношение отдельных уровней иерархии в системе моделирования;

4) целостность отдельных обособленных стадий построения модели. Модель М должна отвечать заданной цели ее создания, поэтому отдельные части должны компоноваться взаимно, исходя из единой системной задачи. Цель может быть сформулирована качественно, тогда она будет обладать большей содержательностью и длительное время может отображать объективные возможности данной системы моделирования. При количественной формулировке цели возникает целевая функция, которая точно отображает наиболее существенные факторы, влияющие на достижение цели. Построение модели относится к числу системных задач, при решении которых синтезируют решения на базе огромного числа исходных данных, на основе предложений больших коллективов специалистов. Использование системного подхода в этих условиях позволяет не только построить модель реального объекта, но и на базе этой модели выбрать необходимое количество управляющей информации в реальной системе, оценить показатели ее функционирования и тем самым на базе моделирования найти наиболее эффективный вариант построения и выгодный режим функционирования реальной системы S .

**5. Этапы моделирования**

****

1. Постановка проблемы и её качественный анализ. Главное на этом этапе - чётко сформулировать сущность проблемы, определить принимаемые допущения, а также определить те вопросы, на которые требуется получить ответ.

Этап включает выделение важнейших черт и свойств моделируемого объекта, основных зависимостей, связывающих его элементы. Здесь же происходит формулирование гипотез, хотя бы предварительно объясняющих поведение объекта.

2. Построение математической модели. Это этап формализации задачи, т.е. выражения ее в виде математических зависимостей и отношений (функций, уравнений, неравенств, схем). Как правило, сначала определяется тип математической модели, а затем уточняются детали.

3. Математический анализ модели. Цель - выявление общих свойств и характеристик модели. Применяются чисто математические приёмы исследования. Наиболее важный момент - доказательство существования решений в сформулированной модели. Если удастся доказать, что задача не имеет решения, то необходимость в последующей работе по данному варианту модели отпадает; следует скорректировать либо постановку задачи, либо способы ее математической формализации.

4. Подготовка исходной информации. Численное моделирование предъявляет жесткие требования к исходной информации. В то же время реальные возможности получения информации существенно ограничивают выбор используемых моделей. При этом принимается во внимание не только возможность подготовки информации (за определенный срок), но и затраты на подготовку соответствующих информационных массивов. Эти затраты не должны превышать эффекта от использования данной информации.

5. Численное решение. Это cоставление алгоритмов, разработка программ и непосредственное проведение расчётов на ЭВМ.

6. Анализ результатов и их применение. На заключительной стадии проверяются правильность, полнота и степень практической применимости полученных результатов.

**6. Теория множеств.**

Как одну из наиболее известных систем графических символов, следует отметить язык диаграмм английского логика Джона Венна. В настоящее время диаграммы Венна применяются для иллюстрации основных теоретико-множественных операций, которые являются предметом специального раздела математики – теории множеств.

Исходным понятием теории множеств является само понятие множество, под которым принято понимать некоторую совокупность объектов, хорошо различимых нашей мыслью или интуицией. При этом не делается никаких предположений ни о природе этих объектов, ни о способе их включения в данную совокупность. Отдельные объекты, составляющие то или иное множество, называют элементами данного множества.

В теории множеств используется специальное соглашение, по которому множества обозначаются прописными буквами латинского алфавита. При этом отдельные элементы обозначаются строчными буквами, иногда с индексами, которые вносят некоторую упорядоченность в последовательность рассмотрения этих элементов. Важно понимать, что какой бы то ни было порядок, вообще говоря, не входит в исходное определение множества. Таким образом, множество, например, квартир 100-квартирного жилого дома с использованием специальных обозначений можно записать следующим образом: A={aj, 02, а3, ..., а{00}. Здесь фигурные скобки служат обозначением совокупности элементов, каждый из которых имеет свой уникальный числовой индекс. Важно понимать, что для данного конкретного множества элемент ato обозначает отдельную квартиру в рассматриваемом жилом доме. При этом вовсе необязательно, чтобы номер этой квартиры был равен 10, хотя с точки зрения удобства это было бы желательно.

Принято называть элементы отдельного множества принадлежащими данному множеству. Данный факт записывается при помощи специального символа "е", который так и называется – символ принадлежности. Например, запись а10ьА означает тот простой факт, что отдельная квартира принадлежит рассматриваемому множеству квартир некоторого жилого дома.

Следующим важным понятием, которое служит прототипом многих более конкретных терминов при моделировании сложных систем, является понятие подмножества. Если есть некоторая совокупность, рассматриваемая как множество, то любая ее часть и будет являться подмножеством. Для обозначения подмножества используется специальный символ. Если утверждается, что множество А является подмножеством множества В, то это записывается как А а В.

Оказывается, тот факт, что некоторая совокупность элементов образует множество, можно обозначить графически в виде круга. В этом случае окружность имеет содержательный смысл или, выражаясь более точным языком, семантику границы данного множества.

Подобным образом можно изобразить и основные теоретикомножественные операции. Так, пересечением двух множеств А и В называется некоторое третье множество С, которое состоит из тех и только тех элементов двух исходных множеств, которые одновременно принадлежат и множеству А, и множеству В. Для этой операции также имеется специальное обозначение: С= А о В.

Следующей операцией, которая также допускает наглядную интерпретацию, является операция объединения множеств. Под объединением двух множеств А и В понимается некоторое третье множество, пусть это будет D, которое состоит из тех и только тех элементов, которые принадлежат или А, или В, или им обоим одновременно. Конечно, специальное обозначение есть и для этой операции: D= AuB.

Понятия мощности множества и отношения множеств. Что касается понятия мощности множества, то данный термин важен для анализа кратности связей, поскольку ассоциируется с количеством элементов отдельного множества. В случае конечного множества ситуация очень простая, поскольку мощность конечного множества равна количеству элементов этого множества. Ситуация усложняется, когда рассматриваются бесконечные множества, т. е. множества, не являющиеся конечными. Не вдаваясь в технические детали, ограничим наше рассмотрение бесконечными множествами счетной мощности. Такими множествами принято считать множества, содержащие бесконечное число элементов, которые, однако, можно перенумеровать натуральными числами 1, 2, 3 и т. д. При этом важно иметь в виду, что достичь последнего элемента при такой нумерации принципиально невозможно, иначе множество окажется конечным.

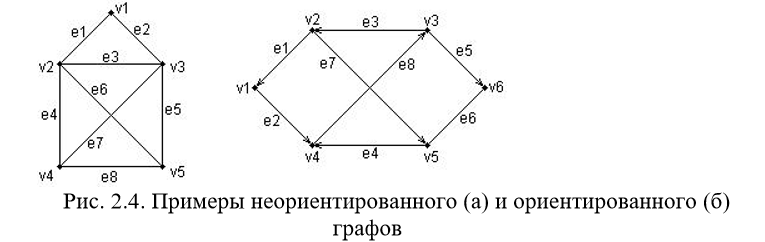
Фундаментальное понятие отношения множеств, которое часто заменяется терминами связь или соотношение. Данный термин ведет свое происхождение от теории множеств и служит для обозначения любого подмножества упорядоченных кортежей, построенных из элементов некоторых исходных множеств. При этом под кортежем понимается просто набор или список элементов, важно только, чтобы они были упорядочены. Другими словами, если рассматривать первый элемент кортежа, то он всегда будет первым в списке элементов, второй элемент кортежа будет вторым элементом в списке и т. д. Можно ли это записать с использованием специальных обозначений? Хотя и существует некоторая неоднозначность в принятых обозначениях, кортеж из двух элементов удобно обозначать как <a1, a2>, из трех элементов – <a1, a2, a3> и т. д. При этом отдельные элементы могут принадлежать как одному и тому же множеству, так и различным множествам. Важно иметь в виду, что порядок выбора элементов для построения кортежей строго фиксирован для конкретной задачи. Речь идет о том, что первый элемент всегда выбирается из первого множества, второй – из второго, и т. д:

Отношение в этом случае будет характеризовать способ или семантику выбора отдельных элементов из одного или нескольких множеств для подобного упорядоченного списка. В этом смысле взаимосвязь является частным случаем отношения. К сожалению, диаграммы Венна не предназначены для иллюстрации отношений в общем случае.

**7. Теория графов**

Графом называется совокупность двух множеств: множества точек или вершин и множества соединяющих их линий или ребер. Формально граф задается в виде двух множеств: G=(V, Е), где V={v1v2, ..., vn} – множество вершин графа, а Е={е1, е2, ..., еm} – множество ребер графа. Натуральное число n определяет общее количество вершин конкретного графа, а натуральное число m – общее количество ребер графа. Следует заметить, в общем случае не все вершины графа могут соединяться между собой, что ставит в соответствие каждому графу некоторое бинарное отношение PQ, состоящее из всех пар вида <vi, vj>, где vi, vj = V. При этом пара <vi, vj> и, соответственно, пара <vj, vi> принадлежат отношению PG в том и только в том случае, если вершины vi и vj соединяются в графе G некоторым ребром ek=Е.

Вершины графа изображаются точками, а ребра – отрезками прямых линий. Рядом с вершинами и ребрами записываются соответствующие номера или идентификаторы, позволяющие их идентифицировать однозначным образом. Ниже представлены два примера конкретных графов (рис. 2.4). При этом первый из них (рис. 2.4, а) является неориентированным графом, а второй (рис. 2.4, б) – ориентированным графом. Как нетрудно заметить, для неориентированного графа ребро е1 соединяет вершины v1 и v2, ребро е2 – вершины v1 и v3, а ребро e3 – вершины v2 и v3 и т. д. Последнее ребро, e8, соединяет вершины v4 и v5, тем самым задается описание графа в целом. Других ребер данный граф не содержит, как не содержит других вершин, не изображенных на рисунке. Так, хотя ребра е6 и e7 визуально пересекаются, но точка их пересечения не является вершиной графа. Для ориентированного графа (рис. 2.4, б) ситуация несколько иная. А именно, вершины v1 и v2 соединены дугой е1, для которой вершина v2 является началом дуги, а вершина v1 – концом этой дуги. Далее дуга е2 соединяет вершины v1 и v4, при этом началом дуги e2 является вершина v1, а концом – вершина v4.



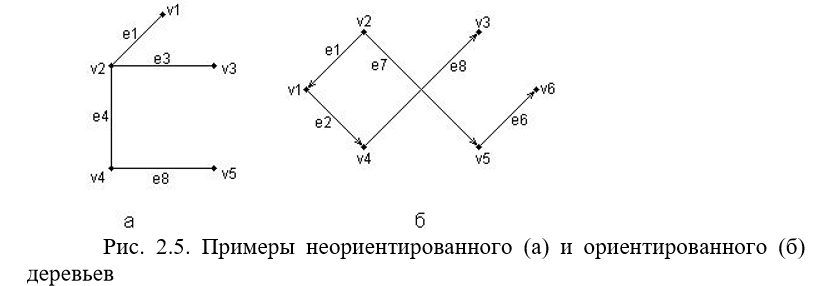
Графы широко применяются для представления различной информации о структуре систем и процессов. Примерами подобных графических моделей могут служить: схемы автомобильных дорог, соединяющих отдельные населенные пункты; схемы телекоммуникаций, используемых для передачи информации между отдельными узлами; схемы программ, на которых указываются варианты ветвления вычислительного процесса.

Общим для всех конкретных подобных моделей является возможность представления информации в графическом виде в форме соответствующего графа. При этом отдельные модели, как правило, обладают дополнительной семантикой и специальными обозначениями, характерными для той или иной предметной области.

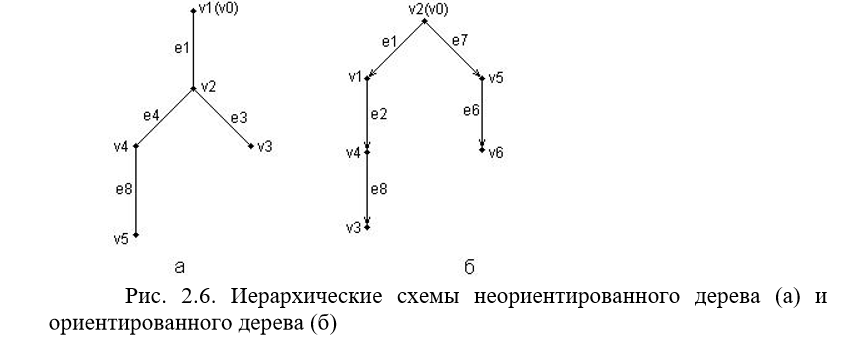
Важными понятиями теории графов являются понятия маршрута и пути, которые ассоциируются с последовательным перемещением от вершины к вершине по соединяющим их ребрам или дугам. Для неориентированного графа маршрут определяется как конечная или бесконечная упорядоченная последовательность ребер S=<, esl, es2, ..., esk>>, таких, что каждые два соседних ребра имеют общую вершину. Нас будут интересовать только конечные маршруты S=<es1, es2, ..., esk>, т. е. такие маршруты, которые состоят из конечного числа ребер. При этом ребро esl принято считать началом маршрута S, а ребро esk – концом маршрута S.

Для ориентированного графа соответствующая последовательность дуг S=<es1, es2, ..., esk> называется ориентированным маршрутом, если две соседние дуги имеют общую вершину, которая является концом предыдущей и началом последующей дуги. Если в маршруте не повторяются ни ребра, ни вершины, то такой неориентированный маршрут называется простой цепью. Если в ориентированном маршруте не повторяются ни ребра, ни вершины, то такой ориентированный маршрут называется путем.

Последнее понятие также иногда применяется для обозначения простой цепи в неориентированных графах и для определения специального класса графов, так называемых деревьев. В общем случае деревья служат для графического представления иерархических структур или иерархий, занимающих важное место в ООАП. Деревом в теории графов называется такой граф D=<V, E>, между любыми двумя вершинами которого существует единственная простая цепь, т. е. неориентированный маршрут, у которого вершины и ребра не повторяются. Ориентированный граф D=<V, Е> называется ориентированным деревом или сокращенно – деревом, если между корнем дерева v0 и любой другой вершиной существует единственный путь, берущий начало в v0. В случае неориентированного дерева (рис. 2.5, а) любая из вершин графа может быть выбрана в качестве корня. Подобный выбор определяется

специфическими особенностями решаемой задачи. Так, вершина v1 может рассматриваться в качестве корня неориентированного дерева, поскольку между v1 и любой другой вершиной дерева всегда существует единственная простая цепь по определению (или, что менее строго, единственный неориентированный путь).

Для случая ориентированного дерева (рис. 2.5, б) вершина v2 является единственным его корнем и имеет специальное обозначение v0. Единственность корня в ориентированном дереве следует из того факта, что ориентированный путь всегда имеет единственную вершину, которая является его началом. Поскольку в теории графов имеет значение только наличие или отсутствие связей между отдельными вершинами, деревья, как правило, изображаются специальным образом в виде иерархической структуры. При этом корень дерева изображается самой верхней вершиной в данной иерархии. Далее следуют вершины уровня 1, которые связаны с корнем одним ребром или одной дугой. Следующий уровень будет иметь номер 2, поскольку соответствующие вершины должны быть связаны с корнем двумя последовательными ребрами или дугами. Процесс построения иерархического дерева продолжается до тех пор, пока не будут рассмотрены вершины, которые не связаны с другими вершинами, кроме рассмотренных, или из которых не выходит ни одна дуга. В этом случае самые нижние вершины иногда называют листьями дерева. Важно иметь в виду, что в теории графов дерево «растет» вниз, а не вверх, как в реальной жизни.

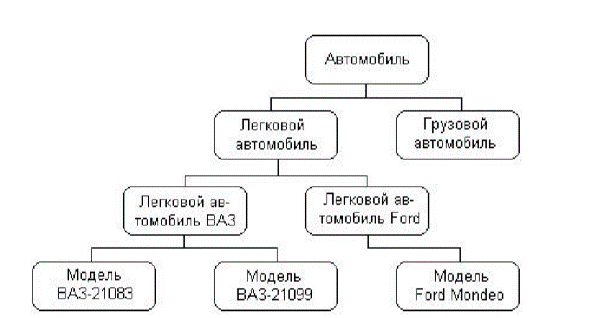
Изображенные выше деревья (рис. 2.5) можно преобразовать к виду иерархий. Например, неориентированное дерево (рис. 2.5, а) может быть представлено в виде иерархического дерева следующим образом (рис. 2.6, а). В этом случае корнем иерархии является вершина v1. Ориентированное дерево (рис. 2.5, б) также может быть изображено в форме иерархического дерева (рис. 2.6, б), однако такое представление является единственным. В первом случае (рис. 2.6, а) вершина v2 образует первый уровень иерархии, вершины v4 и v3 – второй уровень иерархии, вершина v5 – третий и последний уровень иерархии. При этом листьями данного неориентированного дерева являются вершины v3 и v5. Во втором случае (рис. 2.6, б) вершины v1 и v5 образуют первый уровень иерархии, вершины v4 и v6 – второй уровень иерархии, вершина v3 – третий и последний уровень иерархии. Листьями данного ориентированного дерева являются вершины v3 и v6.

**8. Семантические сети**

В общем случае под семантической сетью понимают некоторый граф Gs= =(Vs, Es), в котором множество вершин Vs и множество ребер Es разделены на отдельные типы, обладающие специальной семантикой, характерной для той или иной предметной области. В данной ситуации множество вершин может соответствовать объектам или сущностям рассматриваемой предметной области и иметь вместо номеров вершин соответствующие явные имена этих сущностей. Подобные имена должны позволять однозначно идентифицировать соответствующие объекты, при этом общих формальных правил записи имен не существует. Множество ребер также делится на различные типы, которые соответствуют различным видам связей между сущностями рассматриваемой предметной области.

Так, при построении семантической сети для представления знаний о рабочем персонале некоторой компании в качестве объектов целесообразно выбрать отдельных сотрудников, каждого из которых идентифицировать собственным именем и фамилией. Дополнительно в сети могут присутствовать такие объекты, как рабочие проекты и подразделения компании. В качестве семантических связей можно выделить такие виды, как должностное подчинение сотрудников, участие сотрудников в работе над проектами, принадлежность сотрудников тому или иному подразделению компании.

Важной особенностью семантических сетей является разработка специальных графических обозначений для представления отдельных типов вершин и ребер. При этом вершины не изображаются, как ранее – точками, а имеют вид прямоугольников, овалов, окружностей и других геометрических фигур, конкретный вид которых определяет тот или иной тип сущностей предметной области. Более разнообразным становится и изображение ребер, приобретающих вид различных линий со стрелками или без них, а также имеющих специальные обозначения или украшения в виде условных значков. Соответствующая система обозначений, предназначенная для представления информации об отдельных аспектах моделируемой предметной области, получила название графической нотации.

В качестве конкретного варианта представления информации в виде семантической сети рассмотрим дальнейшее развитие примера с классом «Автомобиль» из главы 1. Фрагмент семантической сети, которая описывает иерархию классов данной предметной области, может быть изображен следующим образом (рис. 2.7). 

На данном рисунке отдельные вершины семантической сети изображаются прямоугольниками с закругленными концами и служат для условного обозначения классов данной предметной области. Соединяющие вершины ребра имеют вполне определенный смысл или семантику. А именно, они явно указывают, что вершина или класс, расположенные на рисунке ниже, являются подклассом того класса уровнем выше, с которым имеется связь в форме соединяющего их ребра. Например, классы «Легковой автомобиль» и «Грузовой автомобиль» являются подклассами класса «Автомобиль», а классы «Модель ВАЗ-21083» и «Модель ВАЗ-21099» являются подклассами класса «Легковой автомобиль производства ВАЗ». Ребра или связи данной семантической сети имеют единственный тип, определяемый семантикой включения классов друг в друга. Поэтому никаких дополнительных обозначений они не содержат.

**9. Методология структурного анализа и моделирования**

Появление первых электронных вычислительных машин или компьютеров ознаменовало новый этап в развитии техники вычислений. Казалось, достаточно разработать последовательность элементарных действий, каждое из которых преобразовать в понятные компьютеру инструкции, и любая вычислительная задача может быть решена. Эта идея оказалась настолько жизнеспособной, что долгое время доминировала над всем процессом разработки программ. Появились специальные языки программирования, которые позволили преобразовывать отдельные вычислительные операции в соответствующий программный код.

Основой данной методологии разработки программ являлась процедурная или алгоритмическая организация структуры программного кода. Это было настолько естественно для решения вычислительных задач, что ни у кого не вызывала сомнений целесообразность такого подхода. Исходным понятием этой методологии являлось понятие алгоритма, под которым, в общем случае, понимается некоторое предписание выполнить точно определенную последовательность действий, направленных на достижение заданной цели или решение поставленной задачи. Примерами алгоритмов являются хорошо известные правила нахождения корней квадратного уравнения или корней линейной системы уравнений.

С этой точки зрения вся история математики тесно связана с разработкой тех или иных алгоритмов решения актуальных для своей эпохи задач. Более того, само понятие алгоритма стало предметом соответствующей теории – теории алгоритмов, которая занимается изучением общих свойств алгоритмов. Со временем содержание этой теории стало настолько абстрактным, что соответствующие результаты понимали только специалисты. Как дань этой традиции какой-то период времени языки программирования назывались алгоритмическими, а первое графическое средство документирования программ получило название блок-схемы алгоритма. Соответствующая система графических обозначений была зафиксирована в ГОСТ 19.701-90, который регламентировал использование условных обозначений в схемах алгоритмов, программ, данных и систем.

Однако потребности практики не всегда требовали установления вычислимости конкретных функций или разрешимости отдельных задач. В языках программирования возникло и закрепилось новое понятие процедуры, которое конкретизировало общее понятие алгоритма применительно к решению задач на компьютерах. Так же, как и алгоритм, процедура представляет собой законченную последовательность действий или операций, направленных на решение отдельной задачи. В языках программирования появилась специальная синтаксическая конструкция, которая получила название процедуры.

Со временем разработка больших программ превратилась в серьезную проблему и потребовала их разбиения на более мелкие фрагменты. Основой для такого разбиения как раз и стала процедурная декомпозиция, при которой отдельные части программы или модули представляли собой совокупность процедур для решения некоторой совокупности задач. Главная особенность процедурного программирования заключается в том, что программа' всегда имеет начало во времени или начальную процедуру (начальный блок) и окончание (конечный блок). При этом вся программа может быть представлена визуально в виде направленной последовательности графических примитивов или блоков (рис. 1.1).

Важным свойством таких программ является необходимость завершения всех действий предшествующей процедуры для начала действий последующей процедуры. Изменение порядка выполнения этих действий даже в пределах одной процедуры потребовало включения в языки программирования специальных условных операторов типа if-then-eise и Goto для реализации ветвления вычислительного процесса в зависимости от промежуточных результатов решения задачи.

Рассмотренные идеи способствовали становлению некоторой системы взглядов на процесс разработки программ и написания программных кодов, которая Получила название методологии структурного программирования. Основой данной методологии является процедурная декомпозиция программной системы и организация отдельных модулей в виде совокупности выполняемых процедур. В рамках данной методологии получило развитие нисходящее проектирование программ или программирование «сверху-вниз». Период наибольшей популярности идей структурного программирования приходится на конец 70-х-начало 80-х годов.

Как вспомогательное средство структуризации программного кода было рекомендовано использование отступов в начале каждой строки, которые должны выделять вложенные циклы и условные операторы. Все это призвано способствовать пониманию или читабельности самой программы. Данное правило со временем было реализовано в современных инструментариях разработки программ.

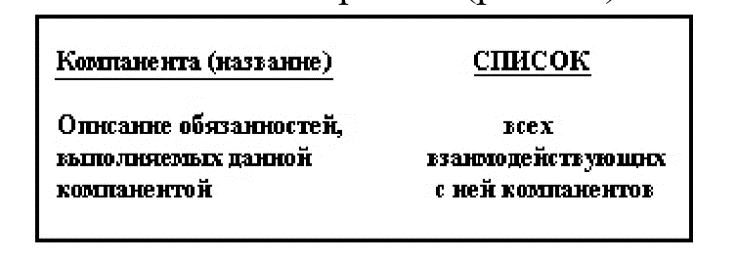
В этот период основным показателем сложности разработки программ считали ее размер. Вполне серьезно обсуждались такие оценки сложности программ, как количество строк программного кода. Правда, при этом делались некоторые предположения относительно синтаксиса самих строк, которые должны были удовлетворять определенным правилам. Общая трудоемкость разработки программ оценивалась специальной единицей измерения – «человеко-месяц» или «человеко-год». А профессионализм программиста напрямую связывался с количеством строк программного кода, который он мог написать и отладить в течение, скажем, месяца.

**10. Методология объектно-ориентированного анализа и проектирования**

Необходимость анализа предметной области до начала написания программы была осознана давно при разработке масштабных проектов.

Процесс разработки баз данных существенно отличается от написания программного кода для решения вычислительной задачи. Главное отличие заключается в том, что при проектировании базы данных возникает необходимость в предварительной разработке концептуальной схемы, которая отражала бы общие взаимосвязи предметной области и особенности организации соответствующей информации. При этом под предметной областью принято понимать ту часть реального мира, которая имеет существенное значение или непосредственное отношение к процессу функционирования программы. Другими словами, предметная область включает в себя только те объекты и взаимосвязи между ними, которые необходимы для описания требований и условий решения некоторой задачи.

Выделение исходных или базовых компонентов предметной области, необходимых для решения той или иной задачи, представляет, в общем случае, нетривиальную проблему. Сложность данной проблемы проявляется в неформальном характере процедур или правил, которые можно применять для этой цели. Более того, такая работа должна выполняться совместно со специалистами или экспертами, хорошо знающими предметную область. Например, если разрабатывается база данных для обслуживания пассажиров крупного аэропорта, то в проектировании концептуальной схемы базы данных должны принимать участие штатные сотрудники данного аэропорта. Эти сотрудники должны хорошо знать весь процесс обслуживания пассажиров или данную предметную область.

Для выделения или идентификации компонентов предметной области было предложено несколько способов и правил. Сам этот процесс получил название концептуализации предметной области. При этом под компонентой понимают некоторую абстрактную единицу, которая обладает функциональностью, т. е. может выполнять определенные действия, связанные с решением поставленных задач. На предварительном этапе концептуализации рекомендуется использовать так называемые CRCкарточки (Component, Responsibility, Collaborator– компонента, обязанность, сотрудники) [1]. Для каждой выделенной компоненты предметной области разрабатывается собственная CRC-карточка (рис. 1.6).

Появление методологии ООАП потребовало, с одной стороны, разработки различных средств концептуализации предметной области, а с другой – соответствующих специалистов, которые владели бы этой методологией. На данном этапе появляется относительно новый тип специалиста, который получил название аналитика или архитектора. Наряду со специалистами по предметной области аналитик участвует в построении концептуальной схемы будущей программы, которая затем преобразуется программистами в код. При этом отдельные компоненты выбираются таким образом, чтобы при последующей разработке их было удобно представить в форме классов и объектов. В этом случае немаловажное значение приобретает и сам язык представления информации о концептуальной схеме предметной области.

Разделение процесса разработки сложных программных приложений на отдельные этапы способствовало становлению концепции жизненного цикла программы. Под жизненным циклом (ЖЦ) программы понимают совокупность взаимосвязанных и следующих во времени этапов, начиная от разработки требований к ней и заканчивая полным отказом от ее использования. Стандарт ISO/IEC 12207, хотя и описывает общую структуру ЖЦ программы, не конкретизирует детали выполнения тех или иных этапов. Согласно принятым взглядам ЖЦ программы состоит из следующих этапов: • Анализа предметной области и формулировки требований к программе

• Проектирования структуры программы

• Реализации программы в кодах (собственно программирования)

• Внедрения программы

• Сопровождения программы

• Отказа от использования программы

На этапе анализа предметной области и .формулировки требований осуществляется определение функций, которые должна выполнять разрабатываемая программа, а также концептуализация предметной области. Эту работу выполняют аналитики совместно со специалистами предметной области. Результатом данного этапа должна являться некоторая концептуальная схема, содержащая описание основных компонентов и тех функций, которые они должны выполнять.

Этап проектирования структуры программы заключается в разработке детальной схемы будущей программы, на которой указываются классы, их свойства и методы, а также различные взаимосвязи между ними. Как правило, на этом этапе могут участвовать в работе аналитики, архитекторы и отдельные квалифицированные программисты. Результатом данного этапа должна стать детализированная схема программы, на которой указываются все классы и взаимосвязи между ними в процессе функционирования программы. Согласно методологии ООАП, именно данная схема должна "служить исходной информацией для написания программного кода.

Этап программирования вряд ли нуждается в уточнении, поскольку является наиболее традиционным для программистов. Появление инструментариев быстрой разработки приложений (Rapid Application Development, RAD) позволило существенно сократить время, и затраты на выполнение этого этапа. Результатом данного этапа является программное приложение, которое обладает требуемой функциональностью и способно решать нужные задачи в конкретной предметной области.

Этапы внедрения и сопровождения программы связаны с необходимостью настройки и конфигурирования среды программы, а также с устранением возникших в процессе ее использования ошибок. Иногда в качестве отдельного этапа выделяют тестирование программы, под которым понимают проверку работоспособности программы на некоторой совокупности исходных данных или при некоторых специальных режимах эксплуатации. Результатом этих этапов является повышение надежности Программного приложения, исключающего возникновение критических ситуаций или нанесение ущерба компании, использующей данное приложение.

Методология ООАП тесно связана с концепцией автоматизированной разработки программного обеспечения (Computer Aided Software Engineering, CASE). Появление первых CASE-средств было встречено с определенной настороженностью. Со временем появились как восторженные Отзывы об их применении, так и критические оценки их возможностей. Причин для столь противоречивых мнений было несколько. Первая из них заключается в том, что ранние CASE-средства были простой надстройкой над некоторой системой управления базами данных (СУБД). Хотя визуализация процесса разработки концептуальной схемы БД имеет немаловажное значение, она не решает проблем разработки приложений других типов.

Вторая причина имеет более сложную природу, поскольку связана с графической нотацией, реализованной в том или ином CASE-средстве. Если языки программирования имеют строгий синтаксис, то попытки предложить подходящий синтаксис для визуального представления концептуальных схем БД были восприняты далеко неоднозначно. Появилось несколько подходов, которые более подробно будут рассмотрены в главе 2. На этом фоне появление унифицированного языка моделирования (Unified Modeling Language, UML), который ориентирован на решение задач первых двух этапов ЖЦ программ, было воспринято с большим оптимизмом всем сообществом корпоративных программистов.

Последнее, на что следует обратить внимание, это осознание необходимости построения предварительной модели программной системы, которую, согласно современным концепциям ООАП, следует считать результатом первых этапов ЖЦ программы. Поскольку язык UML даже в своем названии имеет отношение к моделированию, следует дополнительно остановиться на целом ряде достаточно важных вопросов. Таким образом, мы переходим к теме, которая традиционно не рассматривается в изданиях по ООАП, но имеющая самое прямое отношение к процессу построения моделей и, собственно, моделированию. Речь идет о методологии системного анализа и системного моделирования.

**11. Методология системного анализа и системного моделирования**

Системный анализ как научное направление имеет более давнюю историю, чем ООП и ООАП, и собственный предмет исследования. Центральным понятием системного анализа является понятие системы, под которой понимается совокупность объектов, компонентов или элементов произвольной природы, образующих некоторую целостность. Определяющей предпосылкой выделения некоторой совокупности как системы является возникновение у нее новых свойств, которых не имеют составляющие ее элементы. Примеров систем можно привести достаточно много – это персональный компьютер, автомобиль, человек, биосфера, программа и др. Более ортодоксальная точка зрения предполагает, что все окружающие нас предметы являются системами.

Важнейшими характеристиками любой системы являются ее структура и процесс функционирования. Под структурой системы понимают устойчивую во времени совокупность взаимосвязей между ее элементами или компонентами. Именно структура связывает воедино все элементы и препятствует распаду системы на отдельные компоненты. Структура системы может отражать самые различные взаимосвязи, в том числе и вложенность элементов одной системы в другую. В этом случае принято называть более мелкую или вложенную систему подсистемой, а более крупную – метасистемой.

Процесс функционирования системы тесно связан с изменением ее свойств или поведения во времени. При этом важной характеристикой системы является ее состояние, под которым понимается совокупность свойств или признаков, которые в каждый момент времени отражают наиболее существенные особенности поведения системы.

Рассмотрим следующий пример. В качестве системы представим себе «Автомобиль». Для этого случая система охлаждения двигателя будет являться подсистемой «Автомобиля». С одной стороны, двигатель является элементом системы «Автомобиль». С другой стороны, двигатель сам является системой, состоящей из отдельных компонентов, таких как цилиндры, свечи зажигания и др. Поэтому система «Двигатель» также будет являться подсистемой системы «Автомобиль».

Структура системы «Автомобиль» может быть описана с разных точек зрения. Наиболее общее представление о структуре этой системы дает механическая схема устройства того или иного автомобиля. Взаимодействие элементов в этом случае носит механический характер. Состояние автомобиля можно рассматривать также с различных точек зрения, наиболее общей из которых является характеристика автомобиля как исправного или неисправного. Очевидно, что каждое из этих состояний в отдельных ситуациях может быть детализировано. Например, состояние «неисправный» может быть конкретизировано в состояния «неисправность двигателя», «неисправность аккумулятора», «отсутствие подачи топлива» и пр. Важно иметь четкое представление, что подобная детализация должна быть адекватна решаемой задаче.

Процесс функционирования системы отражает поведение системы во времени и может быть представлен как последовательное изменение ее состояний: Если система изменяет одно свое состояние на другое, то принято говорить, что система переходит из одного состояния в другое. Совокупность признаков или условий изменения состояний системы в этом случае называется переходом. Для системы с дискретными состояниями процесс функционирования может быть представлен в виде последовательности состояний с соответствующими переходами. Более точное графическое описание процесса функционирования систем будет дано в главе 2.

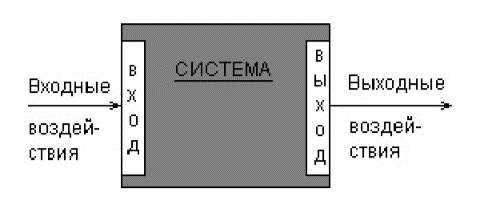
Методология системного анализа служит концептуальной основой системно-ориентированной декомпозиции предметной области. В этом случае исходными компонентами концептуализации являются системы и взаимосвязи между ними. При этом понятие системы является более общим, чем понятия классов и объектов в ООАП. Результатом системного анализа является построение некоторой модели системы или предметной области.

Понятие модели столь широко используется в повседневной жизни, что приобрело очень много смысловых оттенков. Это и «Дом моделей» известного кутюрье, и модель престижной марки автомобиля, и модель политического руководства, и математическая модель колебаний маятника. Применительно к программным системам нас будет интересовать только то понятие модели, которое используется в системном анализе. А именно, под моделью будем понимать некоторое представление о системе, отражающее наиболее существенные закономерности ее структуры и процесса функционирования и зафиксированное на некотором языке или в другой форме.

Примеров моделей можно привести достаточно много. Например, аэродинамическая модель гоночного автомобиля или проектируемого самолета, модель ракетного двигателя, модель колебательной .системы, модель системы электроснабжения региона, модель избирательной компании и др.

Общим свойством всех моделей является их подобие оригинальной системе или системе-оригиналу. Важность построения моделей заключается в возможности их использования для получения информации о свойствах или поведении системы-оригинала. При этом процесс построения и последующего применения моделей для получения информации о системеоригинале получил название моделирование.

Наиболее общей моделью системы является так называемая модель «черного ящика». В этом случае система представляется в виде прямоугольника, внутреннее устройство которого скрыто от аналитика или неизвестно. Однако система не является полностью изолированной от внешней среды, поскольку последняя оказывает на систему некоторые информационные или материальные воздействия. Такие воздействия получили название входных воздействий. В свою очередь, система также оказывает на среду или другие системы определенные информационные или материальные воздействия, которые получили название выходных воздействий. Графически данная модель может быть изображена следующим образом (рис. 1.7).



Ценность моделей, подобных модели «черного ящика», весьма условна. Невольно может возникнуть ассоциация с «Черным квадратом». Однако если оценка изобразительных особенностей последнего не входит в задачи системного анализа, то общая модель системы содержит некоторую важную инфомацию о функциональных особенностях данной системы, которые дают представление о ее поведении. Действительно, кроме самой общей информации о том, на какие воздействия реагирует система, и как проявляется эта реакция на окружающие объекты и системы, другой информации мы получить не можем. В рамках системного анализа разработаны определенные методологические средства, позволяющие выполнить дальнейшую конкретизацию общей модели системы.

Процесс разработки адекватных моделей и их последующего конструктивного применения требует не только знания общей методологии системного анализа, но и наличия соответствующих изобразительных средств или языков для фиксации результатов моделирования и их документирования. Очевидно, что естественный язык не вполне подходит для этой цели, поскольку обладает неоднозначностью и неопределенностью. Для построения моделей были разработаны достаточно серьезные теоретические методы, основанные на развитии математических и логических средств моделирования, а также предложены различные формальные и графические нотации, отражающие специфику решаемых задач. Важно представлять, что унификация любого языка моделирования тесно связана с методологией системного моделирования, т. е. с системой воззрений и принципов рассмотрения сложных явлений и объектов как моделей сложных систем.

Сложность системы и, соответственно, ее модели может быть рассмотрена с различных точек зрения. Прежде всего, можно выделить сложность структуры системы, которая характеризуется количеством элементов системы и различными типами взаимосвязей между этими элементами. Если количество элементов превышает некоторое пороговое значение, которое не является строго фиксированным, то такая система может быть названа сложной. Например, если программная СУБД насчитывает более 100 отдельных форм ввода и вывода информации, то многие программисты сочтут ее сложной. Транспортная система современных мегаполисов также может служить примером сложной системы.

Вторым аспектом сложности является сложность процесса функционирования системы. Это может быть связано как с непредсказуемым характером поведения системы, так и невозможностью формального представления правил преобразования входных воздействий в выходные. В качестве примеров сложных программных систем можно привести современные операционные системы, которым присущи черты сложности как структуры, так и поведения.

**12. Модель IDEF0. Назначение, элементы, принципы построения.**

**Методология IDEF0** основана на подходе разработанном **Дугласом Т. Россом** в начале 70–ых годов и получившем название SADT (Structured Analysis & Design Technique - метод структурного анализа и проектирования). Основу подхода и, как следствие, методологии IDEF0, составляет графический язык описания (моделирования) систем, обладающий следующими свойствами.

* Графический язык - полное и выразительное средство, способное наглядно представлять широкий спектр деловых, производственных и других процессов и операций предприятия на любом уровне детализации.
* Язык обеспечивает точное и лаконичное описание моделируемых объектов, удобство использования и интерпретации этого описания.
* Язык облегчает взаимодействие и взаимопонимание системных аналитиков, разработчиков и персонала изучаемого объекта (фирмы, предприятия), т.е. служит средством «информационного общения» большого числа специалистов и рабочих групп, занятых в одном проекте, в процессе обсуждения, рецензирования, критики и утверждения результатов.
* Язык легок и прост в изучении и освоении.
* Язык может генерироваться рядом инструментальных средств машинной графики;

Методология функционального моделирования IDEF0 – это технология описания системы в целом как множества взаимозависимых действий, или функций.

Важно отметить функциональную направ­ленность IDEF0 – функции системы исследуются независимо от объ­ектов, которые обеспечивают их выполнение. «Функциональная» точка зрения позволяет четко отделить аспекты назначения системы от аспектов ее физической реализации.

IDEF0 методология представляет собой формализованный подход к созданию функциональных моделей – структурных схем изучаемого процесса или системы. Схемы строятся по иерархическому принципу с необходимой степенью подробности.

IDEF0 модели **помогают** разо­браться в том, **что** происходит в изучаемой системе или процессе, какие функции выпол­няются и в какие отношения вступают между собой и с окружающей средой ее функциональные блоки.

IDEF0-модель принципиально **не может** ответить на вопросы о том, **как** протекают процессы в системе во времени и в пространстве.

Наиболее часто IDEF0 применяется как технология исследования и проектирования систем на логическом уровне. По этой причине она, как правило, используется на ранних этапах разработки проекта, для сбора данных и моделирования процесса «как есть». Результаты IDEF0-анализа могут применяться при прове­дении проектирования с использованием моделей для описания потоков работ IDEF3 и потоков данных DFD.

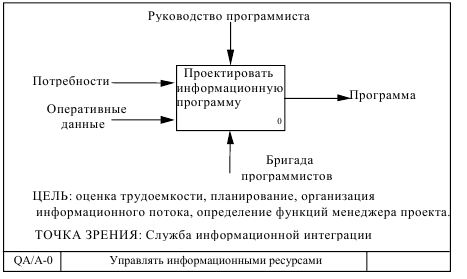
**Элементы модели IDEF0:**

**Диаграммы.**

Модель системы в нотации IDEF0 состоит из набора диаграмм, описывающих различные аспекты функционирования системы.

**Контекстная диаграмма.**

Каждая модель должна иметь контекстную диаграмму верхнего уровня, на которой объект моделирования представлен единственным блоком с граничными стрелками. Эта диаграмма называется A-0 (А минус нуль). Стрелки на этой диаграмме отображают связи объекта моделирования с окружающей средой. Поскольку единственный блок представляет весь объект, его имя – общее для всего проекта. Это же справедливо и для всех стрелок диаграммы, поскольку они представляют полный комплект внешних интерфейсов объекта. Диаграмма A-0 устанавливает область моделирования и ее границу.



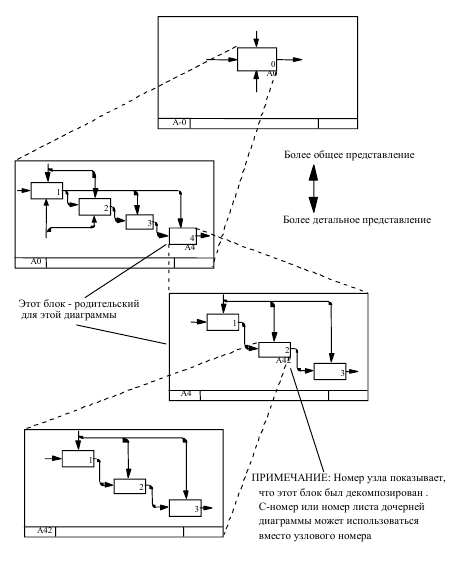
Контекстная диаграмма A-0 также должна содержать краткие утверждения, определяющие **точку зрения** должностного лица или подразделения, с позиций которого создается модель, и **цель**, для достижения которой ее разрабатывают.

Точка зрения определяет, что и в каком разрезе можно увидеть в пределах контекста модели. Изменение точки зрения, приводит к рассмотрению других аспектов объекта.

Формулировка цели выражает причину создания модели, т.е. содержит перечень вопросов, на которые должна отвечать модель, что в значительной мере определяет ее структуру.

**Дочерняя диаграмма.**

Единственная функция, представленная на контекстной диаграмме верхнего уровня, может быть разложена на основные подфункции посредством создания дочерней диаграммы. В свою очередь, каждая из этих подфункций может быть разложена на составные части посредством создания дочерней диаграммы следующего, более низкого уровня, на которой некоторые или все функции также могут быть разложены на составные части.



**Действия.**

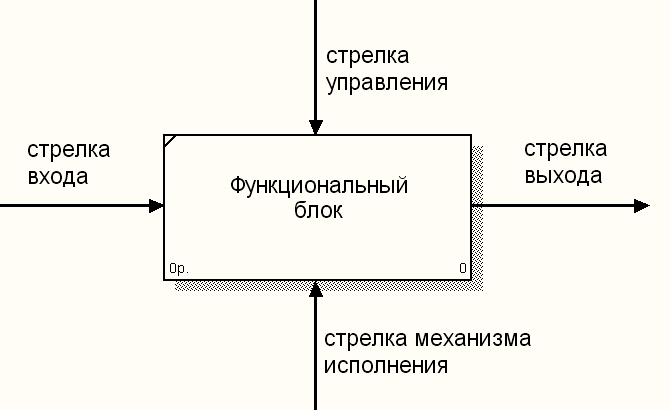
Действие, обычно в IDEF0 называемое функцией, обрабатывает или переводит входные параметры (сырье, информацию и т.п.) в выходные. Поскольку модели IDEF0 представляют систему как множество иерархических функций, в первую очередь должна быть определена функция, описывающая систему в целом – **контекстная функция**. Функции изображаются на диаграммах как **поименованные прямоугольники**, или функциональные блоки. Имена функций в IDEF0 подбираются по сходным правилам с именами действий– с использованием **глаголов или отглагольных существительных**. Важно подбирать имена таким образом, чтобы они отражали систему так, как если бы она обозревалась с точки зрения, выбранной для моделирования.

Каждый блок, не имеющей декомпозиции, помечается небольшой диагональной чертой, расположенной в левом верхнем углу блока.

**Границы и связи**

Связи используются в модели IDEF0 для описания объектов, связанных с выполняемыми функциями. В стандарте IDEF0 существует четыре типа связей:

* I (Input) – вход – нечто, что потребляется в ходе выполнения процесса;
* С (Control) – управление – ограничения и инструкции, влияющие на ход выполнения процесса;
* О (Output) – выход – нечто, являющееся результатом выполнения процесса;
* М (Mechanism) – исполняющий механизм – нечто, что используется для выполнения процесса, но не потребляется само по себе.



**Стрелки входа.**

Вход представляет собой сырье, или информа­цию, потребляемую или преобразуемую функциональным блоком для производства выхода. Стрелки входа всегда направлены в левую сторону прямоугольника, обозначающего в IDEF0 функциональный блок. Наличие входных стрелок на диаграмме не является обязатель­ным, так как возможно, что некоторые блоки ничего не преобразуют и не изменяют. Примером блока, не имеющего входа, может служить «принятие решения руководством», где для принятия решения анализируется несколько факторов, но ни один из них непосредственно не преобразуется и не потребляется в результате принятия какого-либо решения.

**Стрелки управления.**

Стрелки управления отвечают за регулирование того, как и когда выполняется функциональный блок, и, если он выполняется, какой выход получается в результате его выполнения. Так как управление контролирует поведение функционального блока для обеспечения создания желаемого выхода, каждый функ­циональный блок должен иметь, как минимум, одну стрелку управления. Стрелки управления всегда входят в функциональный блок сверху.

Управление можно рассматривать как специфический вид входа.

В случаях, когда неясно, относить ли стрелку к входу или к управле­нию, предпочтительно относить ее к управлению до момента, пока не­ясность не будет разрешена.

**Стрелки выхода.**

Выход – это продукция или информация, по­лучаемая в результате работы функционального блока. Каждый блок должен иметь, как минимум, один выход. Действие, которое не произ­водит никакого четко определяемого выхода, не должно моделиро­ваться вообще (по меньшей мере, должно рассматриваться в качестве одного из первых кандидатов на исключение из модели).

При моделировании непроизводственных предметных областей выходами, как правило, являются данные, в каком-либо виде обрабатываемые функциональным блоком. В этом случае важно, чтобы названия стрелок входа и выхода были достаточно различимы по своему смыслу. Например, блок «Прием пациентов» может иметь стрелку «Данные о пациенте» как на входе, так и на выходе. В такой ситуации входящую стрелку можно назвать «Предварительные данные о пациенте», а исходящую – «Подтвержденные данные о пациенте».

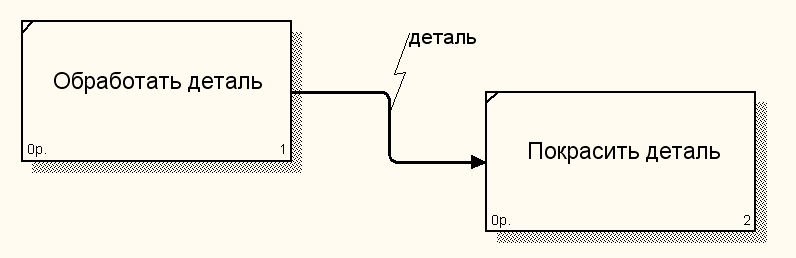
**Стрелки механизма исполнения.**

Механизмы являются ресур­сом, который непосредственно исполняет моделируемое действие.   
С помощью механизмов исполнения могут моделироваться: ключевой персонал, техника и (или) оборудование. Стрелки механизма испол­нения могут отсутствовать в случае, если оказывается, что они не являются необходимыми для достижения поставленной цели моде­лирования.

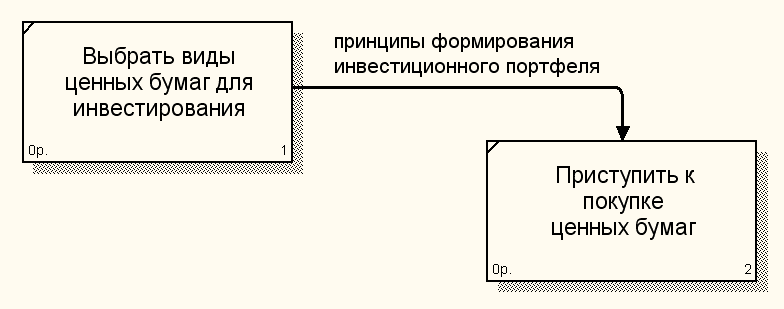
**Комбинированные стрелки.**

В IDEF0 существует пять основных видов комбинированных стрелок:

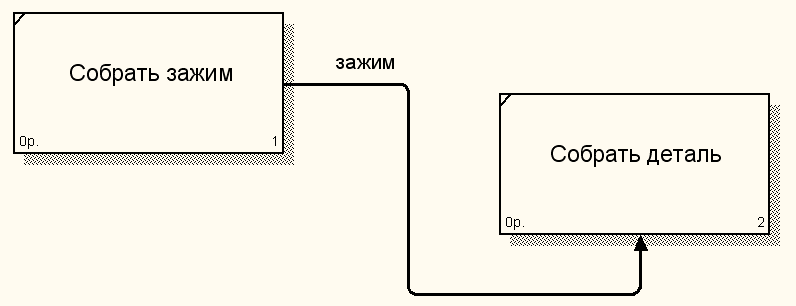
1. Стрелка ***выход – вход*** применяется, когда один из блоков должен полностью завершить работу перед началом работы другого блока .



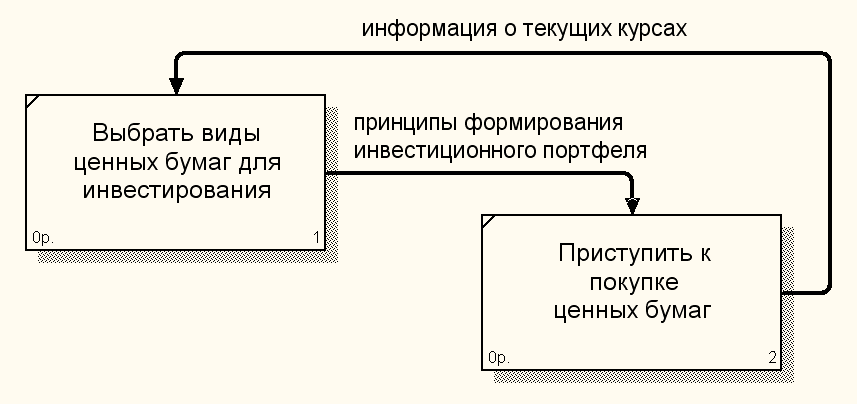
2. Стрелка ***выход − управление*** отражает ситуацию преобладания одного блока над другим, когда один блок управляет работой другого.



3. Стрелки ***выход* − *механизм исполнения*** встречаются реже и отра­жают ситуацию, когда выход одного функционального блока применяется в качестве оборудования для работы другого блока.



4. Стрелка ***выход – обратная связь на управление*** применяется когда зависимые блоки формируют обратные связи для управляющих ими блоков



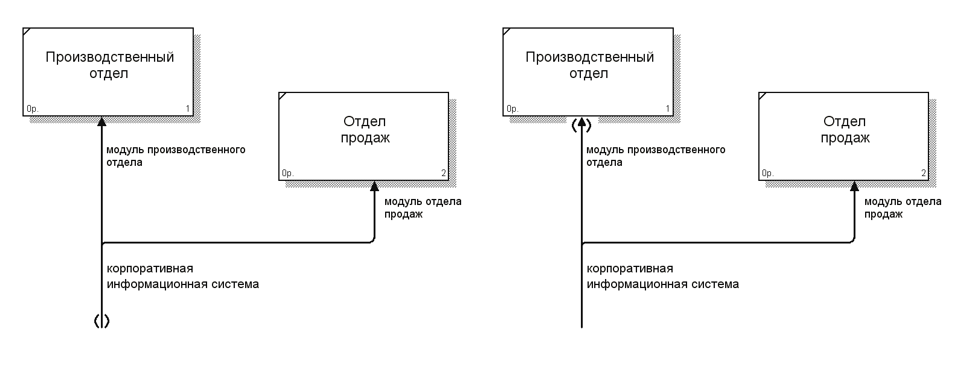
**Ветвление и слияние стрелок.**

Выход функционального блока может использоваться в нескольких других блоках. Фактически чуть ли не главная ценность IDEF0 заключается в том, что эта методология помогает выявить взаимозависимости между блоками системы. Соот­ветственно IDEF0 предусматривает как ветвление, так и слияние стрелок на диаграмме. Разбитые на несколько частей стрелки могут иметь наименования, отличающиеся от наименования исходной стрелки. Исходная и разбитые (или объединенные) стрелки в совокупности называются связанными. Такая техника обычно применяется для того, чтобы отразить использование в процессе только части сырья или информации, обозначаемых исходной стрелкой.

**Туннели.**

Если одна из стрелок диаграммы отсутствует на родительской диаграмме (например, ввиду своей несущественности для родительского уровня) и не связана с другими стрелками той же диаграммы, точка входа этой стрелки на диаграмму или выхода с нее обозначается туннелем, помещается в круглые скобки.

Кроме того, туннели применяются для отражения ситуации, когда стрелка, присутствующая на родительской диаграмме, отсутствует в диаграмме декомпозиции соответствующего блока.



**Процесс построения диаграммы IDEF0:**

1. **Определение целей разработки диаграммы.**

Выбранное определение цели моделирования должно отвечать на следующие вопросы:

– Почему моделируется данный процесс?

– Что выявит данная модель?

– Как ознакомившиеся с этой моделью смогут ее применить?

1. **Выбор точки зрения.**

С методической точки зрения при моделировании полезно использовать мнение экспертов, имеющих разные взгляды на предметную область, однако каждая отдельно взятая модель должна разрабатываться исходя из единственной заранее определенной точки зрения.

Точку зрения нужно подбирать достаточно аккуратно, основой для выбора должна служить поставленная цель моделирования. Наименованием точки зрения может быть наименование должности, подразделения или роли (например, руководитель отдела или менеджер по продажам). Как и в случае с определением цели моделирования, четкое определение точки зрения необходимо для обеспечения внутренней целостности модели и предотвращения постоянного изменения ее структуры.

**3. Определение границ моделирования.**

Границы моделирования имеют два компонента: ширину охвата и глубину детализации. Ширина охвата обозначает внешние границы моделируемой системы. Глубина детализации определяет степень подробности, с которой нужно проводить декомпозицию функциональных блоков.

**4. Создание контекстной диаграммы**

Правила именования блока на контекстной диаграмме не отличаются от именования других блоков диаграммы. Для него обычно подбирают обобщающие названия типа «Управление отделом по работе с клиентами», «Обработка заказов» и т.п.

Стрелки диаграмм IDEF0 обычно проще проектировать в следующем порядке: выход, вход, механизм исполнения, управление.

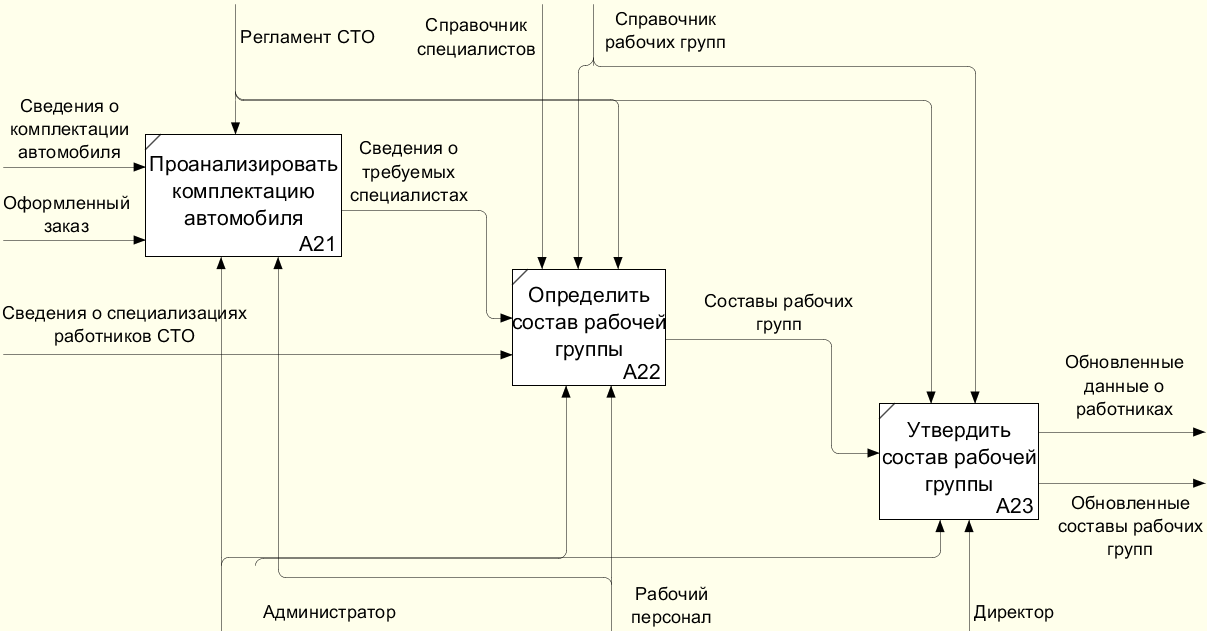
Когда контекстная диаграмма представляется завершенной, попробуйте задать следующие вопросы:

* Обобщает ли диаграмма моделируемый бизнес-процесс?
* Согласуется ли диаграмма с границами моделирования, точкой зрения и целью моделирования?
* Подходит ли выбранный уровень детализации стрелок для контекстного блока? (Обычно на контекстной диаграмме рекомендуется рисовать не более шести стрелок каждого типа.)

**5. Детализация контекстной диаграммы.**

После построения контекстной диаграммы, нужно выделить составные блоки из которых состоит описанный процесс. Они отображаются на диаграмме следующего уровня. На диаграмме должно быть от 3 до 6 блоков. Если выделенный блок представляет собой сложный процесс, то он должен быть разбит на более простые задачи на диаграмме более низкого уровня. При декомпозиции блоков необходимо следить за тем, чтобы не выходить за границы модели.





**14. Моделирование структуры данных с использованием нотации IDEF1X.**

Модель «сущность-связь» была предложена **П. Ченом** (P. Chen) в 1976 году и получила дальнейшее развитие в работах **Р. Баркера** (R. Barker). Диаграммы «сущность-связь» (ERD) предназначены для графического представления моделей данных разрабатываемой программной системы и предлагают некоторый набор стандартных обозначений для определения данных и отношений между ними. С помощью этого вида диаграмм можно описать отдельные компоненты концептуальной модели данных и совокупность взаимосвязей между ними, имеющих важное значение для разрабатываемой системы.

**Элементы модели IDEF1X:**

Согласно стандарту , основными составляющими модели IDEF1X являются:

1. люди, предметы, явления, о которых хранится информация (далее – сущности)
2. связи между этими элементами (далее – отношения)
3. характеристики этих элементов (далее – атрибуты)

Сущность – это множество реальных или абстрактных объектов (людей, мест, событий), обладающих общими атрибутами или характеристиками.

Любой объект системы может быть представлен только одной сущностью, которая должна быть уникально идентифицирована.

Пример. Сущность – Сотрудник. Экземпляр сущности – сотрудник Иванов И.И.

Атрибут – это свойство сущности или связи.

Атрибутам как спецификаторам свойств сущностей или связей присваиваются семантически значимые имена, как правило, в форме существительного.

Атрибутами сущности «Сотрудник» являются: табельный номер, фамилия, имя, отчество, дата рождения и другие персональные характеристики.

Отношения – связь между двумя и более сущностями. Именование отношения осуществляется с помощью грамматического оборота глагола (имеет, определяет, …).

1. Сущность должна иметь **уникальное имя** и именоваться **существительным в единственном числе**.

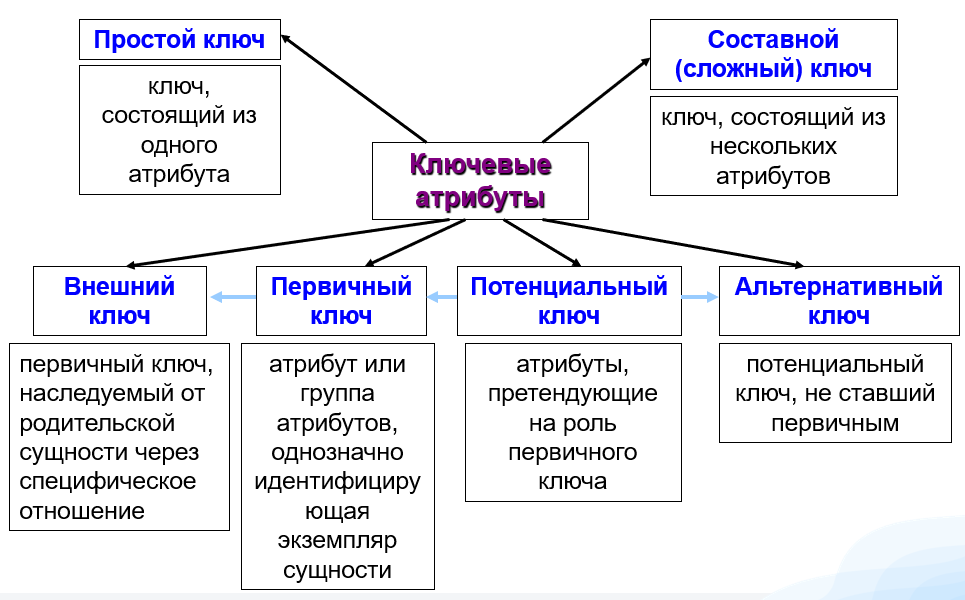
Пример: Студент, Кредитная карта, Договор,…

1. Сущность обладает **одним** или **несколькими атрибутами**, которые ей либо принадлежат, либо наследуются через отношения.
2. Сущность обладает одним или несколькими атрибутами, которые однозначно **идентифицируют** каждый образец сущности и называются **ключом** (составным ключом).
3. Каждая сущность может обладать **любым количеством отношений** с другими сущностями.

В нотации IDEF1X сущность изображается в виде **прямоугольника**. Прямоугольник разделяется на две части горизонтальной линией. Сверху пишутся ключевые атрибуты, снизу остальные атрибуты. Название сущности пишется сверху.



1. Каждый атрибут каждой сущности обладает **уникальным именем.**
2. Сущность может обладать **любым количеством атрибутов.**
3. Различают **собственные** и **наследуемые** атрибуты. Собственные атрибуты являются уникальными в рамках модели. Наследуемые передаются от сущности-родителя при определении идентифицирующей связи.

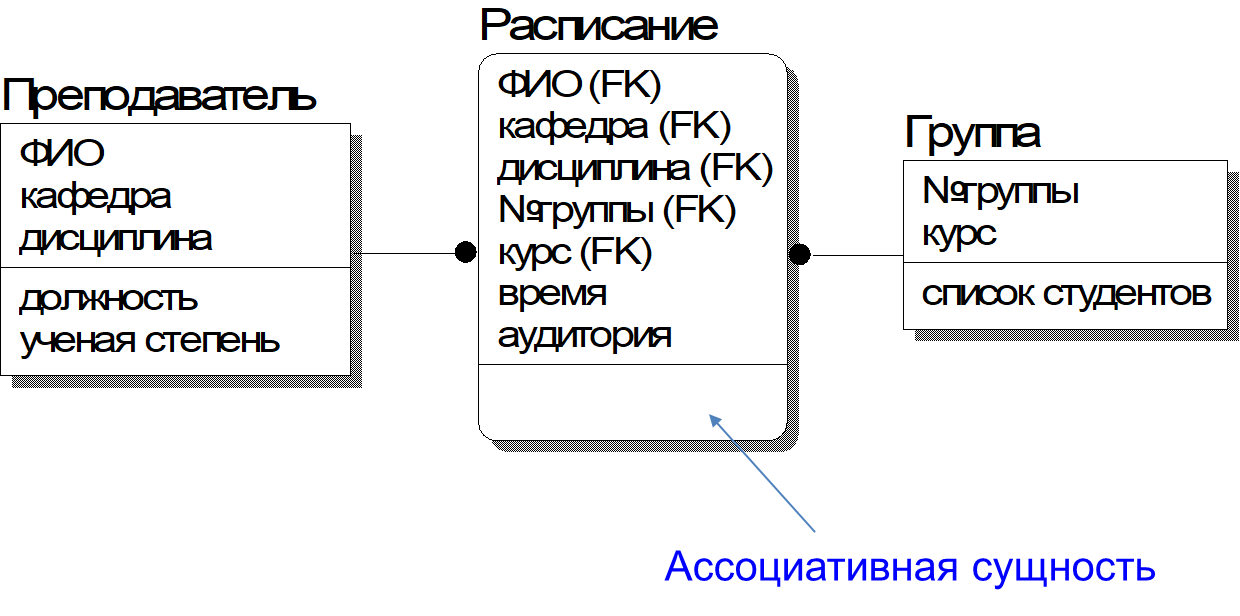
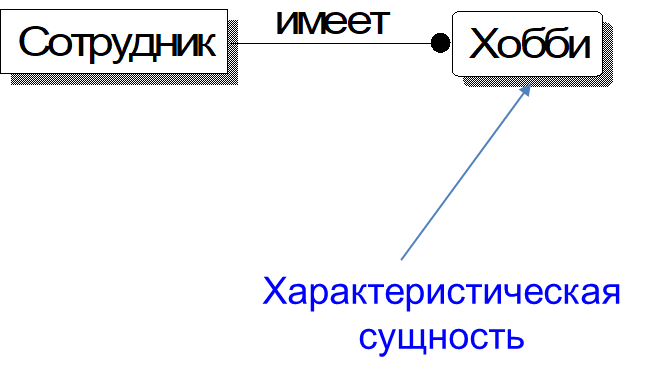


**Типы сущностей**

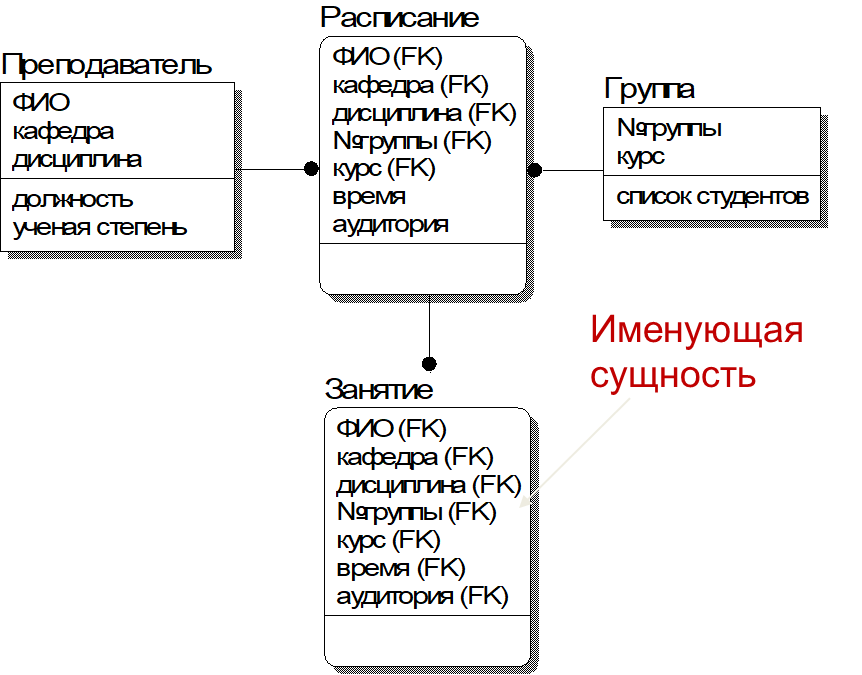
Независимые сущности - представляет собой независимые данные, которые всегда присутствуют в системе, при этом отношения с другими сущностями могут как существовать, так и отсутствовать

Зависимые сущности - представляет данные, зависимые от других сущностей в системе, поэтому она всегда должна иметь отношения с другими сущностями. Зависимые сущности изображаются в виде прямоугольника со скруглёнными краями.

1. **Характеристическая** - это зависимая дочерняя сущность, которая связана только с одной родительской сущностью и по смыслу хранит информацию о характеристиках родительской сущности.
2. **Категориальная** – дочерняя сущность в иерархии наследования
3. **Ассоциативная** - сущность, связанная с несколькими родительскими сущностями. Такая сущность содержит информацию о связях сущности



**4. Именующая** - частный случай ассоциативной сущности, не имеет собственных атрибутов, только атрибуты родительской сущности.

1. При определении отношения типа *«родитель-потомок»*:

1.1. Экземпляр потомка связан с одним родителем

1.2. Экземпляр-родитель может быть связан с несколькими экземплярами потомков.

2. В **идентифицирующем** отношении сущность-потомок всегда является зависимой от идентифицирующей сущности.

**Виды отношений**

**Идентифицирующее отношение.** Сущность А1 однозначно определяет сущность А2. Ее первичный ключ наследуется в качестве первичного ключа сущностью А2 (внешний ключ)

**Неидентифицирующее отношение.** Сущность А1 связана с сущностью А2, но однозначно не определяет ее. Первичный ключ сущности А1 наследуется в качестве неключевого атрибута сущности А2

**Отношение «многие-ко-многим» (неспецифическое).** Сущности А1 и А2 имеют формальную связь, но наследования атрибутов не происходит.

3) Сущность может быть связана с любым количеством других сущностей как в качестве родителя, так и в качестве потомка.

4) Отношение определяется мощностью. *Мощность* связи служит для обозначения отношения количества экземпляров родительской сущности к числу экземпляров дочерней.

а) общий случай, когда одному экземпляру родительской сущности соответствуют **0, 1 или много** экземпляров дочерней сущности

б) когда одному экземпляру родительской сущности соответствует **1 или много** экземпляров дочерней (0 исключается ).

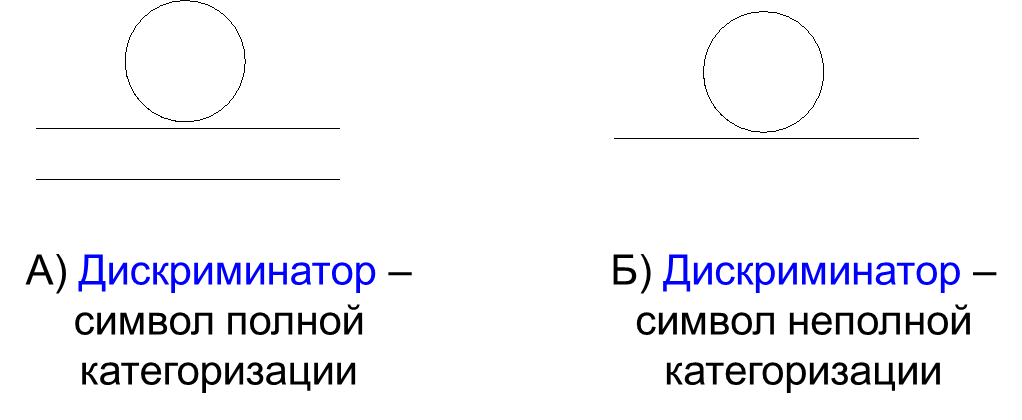
в) когда одному экземпляру родительской сущности соответствует **0 или 1** экземпляр дочерней сущности.

г) когда одному экземпляру родительской сущности соответствует **заранее заданное число** экземпляров дочерней сущности.

Отношения **категоризации** – отношения между двумя и более сущностями, в которых каждый экземпляр одной сущности, называемой общей, связан в точности с одним экземпляром сущности, называемой сущностью-категорией.

Категория выделяется из общей сущности по определенному **признаку.**

Различают **полную** и **неполную** категоризацию





Б) Дискриминатор – символ неполной категоризации

1. Сущность типа «категория» может иметь только одну общую сущность.

2. Сущность-категория, принадлежащая одному отношению категоризации, может быть общей сущностью в другом отношении категоризации.

3. Сущность может являться общей в *любом количестве* отношений категоризации.

4. Атрибуты первичного ключа сущности-категории должны совпадать с атрибутами первичного ключа общей сущности.

5. Все экземпляры сущности-категории имеют одно и то же значение дискриминатора, следовательно, все экземпляры других категорий должны иметь другое значение дискриминатора.



1. Все стрелки (вход, выход, управление, механизм) функциональной модели становятся потенциальными сущностями, а функции, связывающие их, трансформируются в отношения между этими сущностями. Для этого составляется **пул** – список потенциальных сущностей.

2. Число сущностей и связей в IDEF1X-модели считается необозримым, если их количество превышает 25-30. Поэтому далее рассматривается совокупность сущностей и отношений для каждой функции.

3. Информационная модель функции должна позволять воспроизвести структуру документа и часть информации в нем, а также воспроизвести информацию порождаемого документа.

4. Текстовые пояснения заносятся в глоссарий или оформляются гипертекстом.

5. На основании определения типов отношений, анализа функций и дальнейшего изучения предметной области определяются атрибуты.

